Computational Visual Media
DOI 10.1007/s41095-016-0069-x Vol. 3, No. 1, March 2017, 61–71

Research Article

Boundary-aware texture region segmentation from manga

Xueting Liu[1,2], **Chengze Li**[1,2], **and Tien-Tsin Wong**[1,2] (✉)

Abstract Due to the lack of color in manga (Japanese comics), black-and-white textures are often used to enrich visual experience. With the rising need to digitize manga, segmenting texture regions from manga has become an indispensable basis for almost all manga processing, from vectorization to colorization. Unfortunately, such texture segmentation is not easy since textures in manga are composed of lines and exhibit similar features to structural lines (contour lines). So currently, texture segmentation is still manually performed, which is labor-intensive and time-consuming. To extract a texture region, various texture features have been proposed for measuring texture similarity, but precise boundaries cannot be achieved since boundary pixels exhibit different features from inner pixels. In this paper, we propose a novel method which also adopts texture features to estimate texture regions. Unlike existing methods, the estimated texture region is only regarded an initial, imprecise texture region. We expand the initial texture region to the precise boundary based on local smoothness via a graph-cut formulation. This allows our method to extract texture regions with precise boundaries. We have applied our method to various manga images and satisfactory results were achieved in all cases.

Keywords manga; texture segmentation

1 Introduction

Manga is a world-wide popular form of entertainment enjoyed by people of all ages (see Fig. 1). Nowadays, with the development of electronic devices, more and more people read manga on electronic devices such as computers, tablets, and even cellphones. With the power of electronic devices, manga can be presented in a visually enriched form by adding color, motion, or stereoscopic effects. There is thus a rising trend in the manga industry to convert legacy manga books into digital versions. During the digitization process, one major challenge is to segment texture regions from manga as a basis for various applications such as vectorization and colorization. However, texture region segmentation is not easy for manga since textures in manga are composed of lines. This leads to the difficulty of discriminating structural lines from textures, and further leads to the difficulty of identifying the precise boundary of each texture region. Therefore, texture region segmentation is still manually performed in the manga industry currently. As one may imagine, this process is quite tedious and time-consuming.

To help identify and classify textures, various texture features have been proposed in the computer vision field [1]. The similarity of the texture features for two pixels shows whether these two pixels are inside the same texture region. Texture segmentation

Fig. 1 Manga images.

1 The Chinese University of Hong Kong, Hong Kong, China. E-mail: X. Liu, xtliu@cse.cuhk.edu.hk; C. Li, czli@cse.cuhk.edu.hk; T.-T. Wong, ttwong@cse.cuhk.edu.hk (✉).

2 Shenzhen Research Institute, the Chinese University of Hong Kong, Shenzhen, China.

Manuscript received: 2016-09-09; accepted: 2016-12-20

or texture classification techniques can be further applied to extract texture regions based on similarity of texture features. However, since texture features are analyzed within a local neighborhood, pixels in the interior of a texture region (e.g., the blue box in Fig. 2) and pixels near the boundary of a texture region (e.g., the red and orange boxes in Fig. 2) exhibit different texture features (see Fig. 2(c)), even though they belong to the same texture region. Thus, pixels near texture boundaries may be mistakenly regarded as not belonging to that texture region. This will lead to imprecise boundaries of the segmented texture region if only similarity of texture features is considered (see Fig. 3).

To resolve the boundary issue, we noticed that texture smoothing techniques are quite powerful in suppressing local textures while still preserving sharp boundaries. Using texture smoothing methods, one could suggest first smoothing the input manga image, and then performing intensity-based image segmentation to extract texture regions. However, a texture region may have spatial varying textures (e.g., the background region in Fig. 4(a)

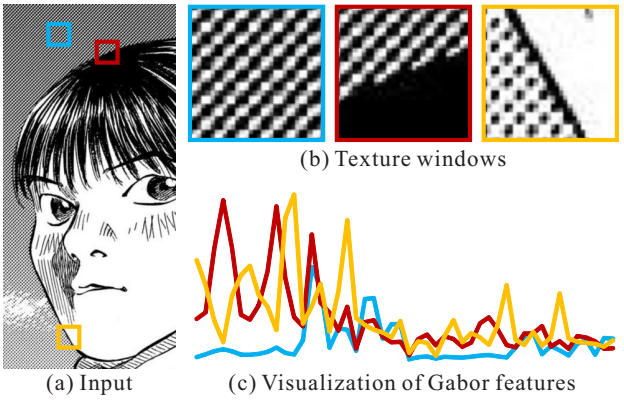


Fig. 2 For each texture region (e.g., the background region), interior pixels (blue box) and boundary pixels (red and orange boxes) exhibit different texture features.

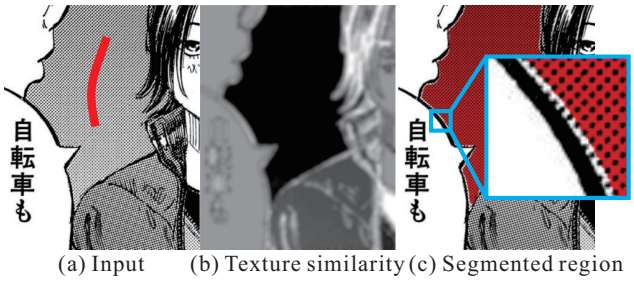


Fig. 3 Texture region segmentation based on Gabor features. Note that a precise boundary cannot be achieved.

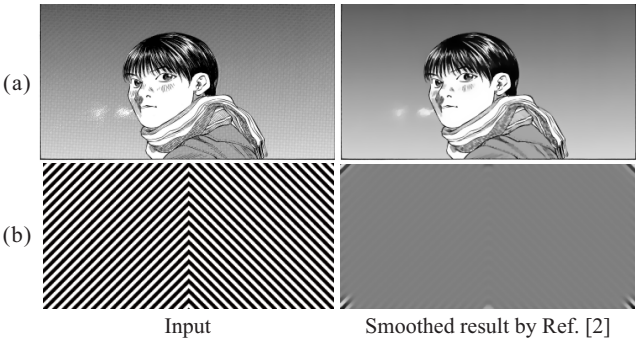


Fig. 4 (a) A background region with a spatial-varying texture. (b) Different textures may have similar intensities after smoothing.

changes from dark to light vertically). Furthermore, the texture smoothing method is also incapable of differentiating textures with similar overall intensities (as in Fig. 4(b)). Therefore, we still need to analyze texture features in order to robustly handle spatial-varying textures and textures with similar intensities.

In this paper, we propose a novel, user-interaction-based texture region segmentation system, which integrates texture feature analysis and texture smoothing techniques in order to segment texture regions with precise boundaries from manga images. The user may draw one or several strokes inside the region or object to be segmented. Our system automatically estimates a region mask from the user input. To do so, we first summarize the texture features of the user-drawn strokes, and estimate an initial texture region having similar texture features to the user-drawn strokes. In this step, we adopt a conservative similarity measurement for estimating the initial texture regions in order to guarantee that all pixels inside the initial texture region lie within the user-specified region. We then expand the initial texture region to the precise boundaries using a graph-cut formulation. We formulate the accumulated smoothness difference from the initial texture region as the data cost, and the local smoothness difference as the smoothness cost. Using this formulation, we can extract the user-specified region with a precise boundary.

We demonstrate the effectiveness of our texture region segmentation system on a set of manga images containing various types of textures, including regular textures, irregular textures, and spatial-varying textures. Our contributions can be summarized as follows:

- We propose a novel user-interaction-based system for extracting texture regions with precise boundaries.
- Our system can handle user strokes drawn across multiple different textures simultaneously.

2 Related work

While only a few existing research work are tailored for extracting texture regions from manga, extensive research has been done on identifying and segmenting textures from natural photos. We can roughly classify this related research into three categories: feature-based texture segmentation, regular texture analysis, and texture smoothing.

Feature-based texture segmentation. Texture features are based on statistical models describing local characteristics of point neighborhoods. The statistical model of a texture feature usually contains a range of texture properties, such as size, aspect ratio, orientation, brightness, and density [3]. Various texture features have been proposed to describe and model textures including various Gabor filters [4], filter bank responses [5], random field models, wavelet representations, and so on. In particular, the Gabor filter was adopted by Ref. [6] to analyze textures in manga, and is still considered to be the state-of-the-art method. Texture features are utilized in various applications such as texture classification, segmentation, and synthesis. Texture segmentation methods fall into two categories, supervised [7] and unsupervised [6, 8]. In particular, Ref. [6] proposed to segment textures in manga images via a level-set method based on Gabor features. However, as we have stated, although feature-based texture segmentation methods can identify textures well and differentiate between textures, precise region boundaries cannot be achieved since boundary pixels exhibit different texture features from interior pixels. In contrast, our method can extract texture regions with precise boundaries.

Regular texture analysis. For regular or near-regular texture patterns, attempts have been made to detect and analyze the regularity of the textures based on spatial relationships [9–11]. In particular, Liuy et al. [12] considered how to detect and remove fences in natural images. A stream of research has also considered de-screening, i.e., detecting and smoothing halftone textures. An in depth survey of de-screening can be found in Ref. [13]. Kopf and Lischinski [14] discussed how to extract halftone patterns in printed color comics by modeling dot patterns. Very recently, Yao et al. [15] considered how to extract textures from manga by modeling three specific texture primitives, dots, stripes, and grids. However, these methods can only handle a small set of pre-defined regular textures. In comparison, our method can handle regular or irregular, and even spatial-varying, textures of the kind that exist in real manga images.

Texture smoothing. In order to differentiate textures and structures, various edge-preserving texture smoothing methods have been proposed, such as the total variation regularizer [16–19], bilateral filtering [20–22], local histogram-based filtering [23], weighted least squares [24], extrema extraction and extrapolation [25], L_0 gradient optimization [26], and relative total variation (RTV) [2]. While these texture smoothing methods may suppress local oscillations based on local information, they are incapable of identifying a texture region or differentiating between two textures. This is because that these methods do not model textures or structures, so they do not have a higher-level understanding of the semantics of the textures. In this paper, we thus utilize texture features to identify textures, but we incorporate texture smoothing techniques to identify sharp texture boundaries.

3 Overview

The input to our system includes a manga image (see Fig. 5(a)) and one or more user-specified strokes (see Fig. 5(b)). To extract regions with similar textures to the ones identified by the user-specified strokes, we first summarize the texture features of the pixels belonging to the strokes. The texture features we use are Gabor features, which are also used by Ref. [6] in a manga colorization application. Since textures may spatially vary, and one user-specified stroke may go across several texture regions (see Fig. 5(b)), we summarize several main texture features inside each stroke by clustering. Then we

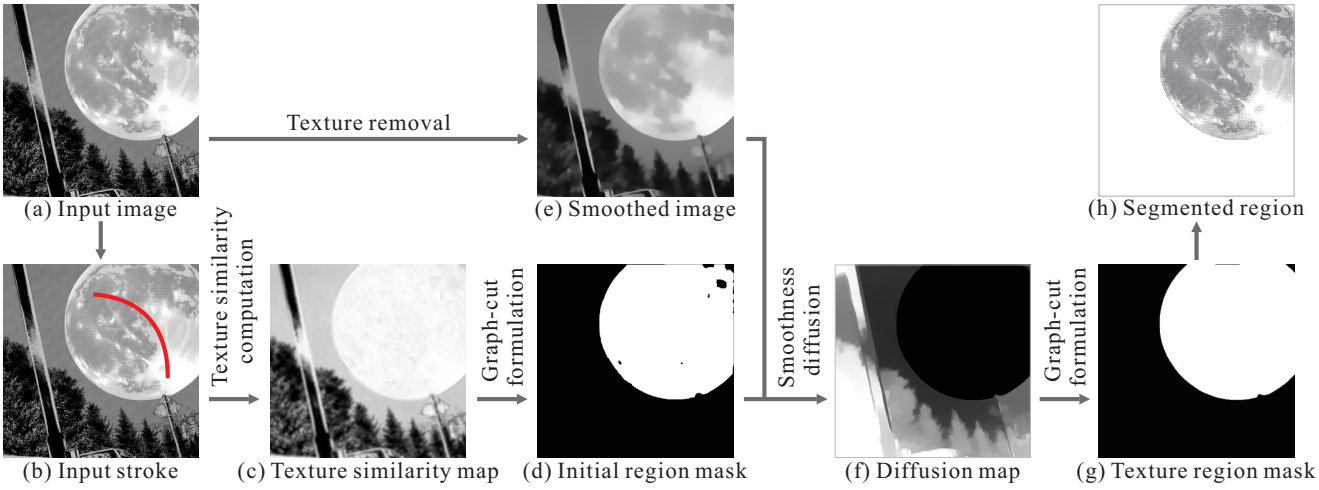


Fig. 5 System overview (image size: 778 × 764, loading time: 3.51 s, processing time: 0.58 s).

calculate the similarity between the texture feature of each pixel in the manga image and the clustered main texture features, to form a texture similarity map (see Fig. 5(c)). In this map, intensity of pixel values indicates similarity of texture feature values with those of the user-specified strokes. Based on the computed texture similarities, we then obtain one or several initial texture regions using a graph-cut formulation (see Fig. 5(d)). Our initial texture region extraction method is detailed in Section 4.

We then expand the initial texture regions to their precise boundaries. To do so, we first obtain a smoothed image from the input image using texture smoothing techniques (see Fig. 5(e)). Amongst all existing texture smoothing techniques, we found that the RTV metric proposed by Ref. [2] is the most effective at smoothing textures while preserving sharp structures in manga images. In the smoothed image, if two neighboring pixels have close intensity values, it means that these two pixels are very likely to be inside the same texture region. Conversely, if two neighboring pixels have a jump in intensity values, it means that these two pixels are very likely to be inside two different regions. Therefore, we can diffuse the initial texture regions using local intensity continuity of the smoothed image to obtain a diffusion map (see Fig. 5(f)). This diffusion map shows how smoothly each pixel is connected to the initial regions. If a pixel has a low diffusion value, it means that this pixel is smoothly connected to the initial region, so it is very likely that this pixel is inside the same texture region. Conversely, a pixel with a high diffusion value is very likely to lie outside the texture region. Finally, we extract the precise texture region based on the diffusion map using another graph-cut formulation (see Figs. 5(g) and 5(h)). Our region expansion method is detailed in Section 5.

We have evaluated our system with various manga, and the results are shown in Section 6. We also show how users can easily adjust the retrieved texture region using a single parameter.

4 Initial region extraction

Given an input manga image, and one or more user-specified strokes, we first extract the initial regions with similar textures to the user-specified strokes. To do so, we first summarize the texture features of the pixels inside the strokes. Then we obtain a texture similarity map which gives the texture similarity between each pixel in the image and the summarized texture features. The initial regions are then extracted using a graph-cut formulation.

4.1 Texture feature summarization

To judge whether two pixels have similar textures, we use statistical features in the Gabor wavelet domain [27], which have already proved useful in differentiating textures in manga [6]. A Gabor feature vector is an $M \times N$-dimensional vector where M is the number of scales and N is the number of orientations used in the Gabor feature analysis. In this paper, we fix the numbers of scales and orientations to $M = 4$ and $N = 6$ respectively in all our experiments. Therefore, for a pixel p in the

manga image, its Gabor feature vector $\boldsymbol{G}_p$ describing local texture feature around p is 24-dimensional.

Given a set of user-specified strokes $U = \{u_1, u_2, \dots\}$, we could calculate a main texture feature for these strokes by averaging the texture features of all pixels inside the strokes as $\sum_{p\in U} \boldsymbol{G}_p/|U|$ where $|U|$ is the cardinality of U. However, a textured area may have a spatial-varying texture, or a single user-specified stroke may also go across multiple textured areas at the same time. For example, the moon in Fig. 5 cannot be specified by a single texture feature vector. Therefore, we represent the textures determined by the user-specified strokes using multiple texture feature vectors. To extract the most representative textures for the user-specified strokes, we use the k-means clustering method to cluster the texture features of all pixels inside the strokes into k groups as $\boldsymbol{T}_U = \{\boldsymbol{T}_1, \dots, \boldsymbol{T}_k\}$. These satisfy:

$$\underset{\boldsymbol{T}_U}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{p\in U} \sum_{\boldsymbol{G}_p \in \boldsymbol{T}_i} \|\boldsymbol{G}_p - \boldsymbol{T}_i\|_2 \tag{1}$$

where $\|\cdot\|_2$ is the L_2-norm operator. In our experiments, a single textured area can usually be represented by 1–5 texture feature vectors. By setting k to a higher value we can ensure that the representative texture features are descriptive enough for a spatial-varying texture region or multiple texture regions in a single stroke. We empirically set $k = 20$ in all our experiments to balance descriptiveness and effectiveness.

4.2 Initial region extraction via graph-cut

From the summarized representative texture feature vectors of the user-specified strokes, we then calculate the texture similarity value between each pixel p and the representative textures $\boldsymbol{T}_U$ as

$$C(p) = \min_i \|\boldsymbol{G}_p - \boldsymbol{T}_i\|_2 \tag{2}$$

Pixels with higher texture similarity are more likely to be inside the texture region specified by the user.

Using the calculated texture similarity, we extract an initial texture region via a graph-cut formulation. There are two terminal nodes, the source and the sink, in our graph. Each pixel p in the image corresponds to a non-terminal node n_p, which is connected to both source and sink. If the graph-cut result connects a non-terminal node (pixel) to the source, it means this pixel lies inside the initial regions. Otherwise, this pixel lies outside the initial regions. Each edge connecting a non-terminal node and a terminal node is associated with a data cost:

$$D(n_p, \text{source}) = \widetilde{C}(p) \tag{3}$$

$$D(n_p, \text{sink}) = 1 - \widetilde{C}(p) \tag{4}$$

where $\widetilde{C}(p)$ is the normalized value of $C(p)$. Intuitively speaking, if the texture of a pixel p is very similar to that of one of the representative texture features, $D(n_p, \text{source}) = C_p$ should be relatively high, and therefore it is very likely that p will be connected to the source, i.e., inside the initial region, after the graph-cut.

For every pair of (4-connected) neighboring pixels p and q in the image, we connect n_p and n_q by an edge. Each edge connecting two non-terminal nodes is associated with a smoothness cost which measures our confidence that these two neighboring pixels should be assigned the same label, and therefore belong to the same texture region. We model the smoothness cost as the magnitude of the difference of the texture feature vectors of these two nodes n_p and n_q:

$$S(n_p, n_q) = \|\boldsymbol{G}_p - \boldsymbol{G}_q\|_2 \tag{5}$$

Intuitively speaking, if two neighboring pixels p and q have similar textures, the smoothness cost $S(n_p, n_q)$ should be low, and there is high probability for them to be assigned the same label, and so belong to the same texture region.

After constructing the graph, we can obtain the optimal cut through an optimization process which minimizes the energy function:

$$\sum_{p,u} D(n_p, u) + w_{\text{c}} \sum_{p,q} S(n_p, n_q) \tag{6}$$

where $u \in \{\text{source}, \text{sink}\}$ is the label, and w_{c} weights the data cost and smoothness cost. We experimentally set w_{c} to 1 in all our experiments. The pixels labeled as source after graph-cut form the initial regions. Since other regions might also have similar patterns to the user-specified region, we remove regions that do not intersect the user-specified strokes. For regions that contain spatial-varying textures, different strokes may lead to different initial regions (see Fig. 6) and affect the followed expansion. The user-specified strokes should go across all different textures in order to achieve good segmentation results.

5 Initial region expansion

Starting from the extracted initial regions, we now

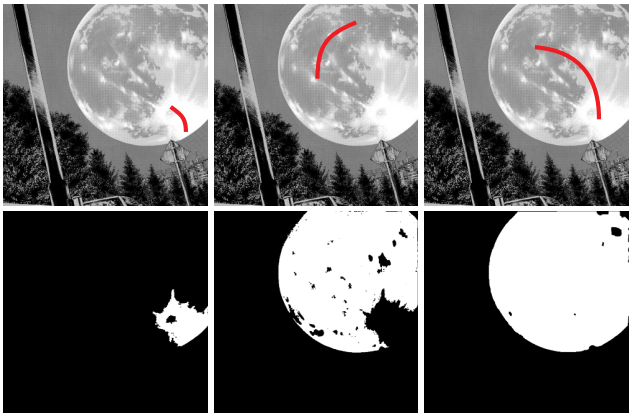


Fig. 6 Initial region extraction from strokes. Top: several user-specified strokes. Bottom: corresponding extracted initial regions.

show how we expand the regions to their precise boundaries. In short, we first smooth the input image and diffuse the initial regions based on the smoothed image. Then we extract the final texture region with precise boundary via another graph-cut formulation.

5.1 Smoothness map construction

To smooth a manga image, we have experimentally determined that the relative total variation method proposed by Xu et al. [2] performs best among the large pool of existing methods. This is mainly because that this measurement is more tolerant of high-contrast textures than other methods. This method has two input parameters λ and σ. Here, λ is a weighting factor which we set to 0.015 in all our experiments. σ is the local window size for measuring local oscillations (textures). In our experiments, we found that $\sigma = 5$ works best for most textures. But when the texture is sparser, we may need to assign σ a higher value. Some texture smoothing results of manga images are shown in Figs. 4 and 5(e).

In the smoothed image, if two neighboring pixels have similar intensity values, it is very likely that they are in the same texture region, and vice versa. We also observe that texture regions in manga images are usually enclosed by black boundary lines. Therefore, we can judge whether a pixel is likely to be inside the user-specified region by measuring whether this pixel is *smoothly connected* to the initial regions. Here, by saying smoothly connected, we mean that there exists a path from this pixel to the initial regions where the intensity values of the pixels change smoothly along the path. Formally, given the initial regions R and a pixel p outside the initial region, we define a path from R to p as a set of pixels $h(R,p) = \{q_1, \ldots, q_l\}$ where $\|q_i - q_{i+1}\|_1 \leqslant 1$, $q_1 \in R$ and $q_L = p$. Here, $\|\cdot\|_1$ is the L_1-norm operator. We can measure whether p is smoothly connected to R along a path $h(R,p)$ by accumulating the intensity differences along this path:

$$v_h(R,p) = \sum_{i=1}^{l-1} \left| J_{q_{i+1}} - J_{q_i} \right| \tag{7}$$

where J is the smoothed manga image. Since there is more than one path from p to R, we can measure whether p is smoothly connected to R by taking the minimal smoothness value of all possible paths:

$$v_H(R,p) = \min_{h(R,p) \in H(R,p)} v_h(R,p) \tag{8}$$

In a practical implementation, we compute the above smoothness values via a diffusion process. More concretely, we first construct a diffusion map F by setting pixels inside the initial regions to 0 and pixels outside initial regions to $+\infty$. Then we iteratively update the smoothness value of each pixel based on its surrounding pixels using:

$$F_p = \min \left\{ F_p, \min_{\|q-p\|_1 \leqslant 1} F_q + |J_p - J_q| \right\} \tag{9}$$

We visualize the diffusion process in Fig. 7.

5.2 Final region extraction via graph-cut

While we could extract a final texture region by thresholding the diffusion map, we have found that naive thresholding generally leads to bumpy and leaky boundaries. To avoid these issues, we formulate another graph to extract the final region. As in the previous graph-cut formulation, each pixel p is formulated as a non-terminal node n_p, and is connected to two terminal nodes, the source and the sink. If the graph-cut result labels a non-terminal node (pixel) as connected to the source, it means this pixel is inside the final texture region; otherwise, it is outside. The data cost associated with each edge connecting a terminal node and a non-terminal node measures how likely this pixel is smoothly connected to the initial region based on the smoothness map, and is expressed as

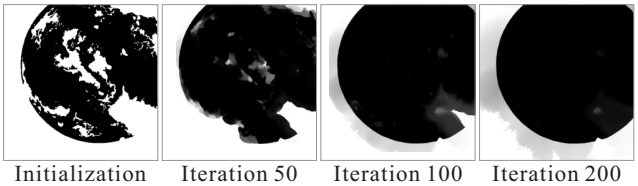


Fig. 7 The diffusion process.

$$\widetilde{D}(n_p, \text{source}) = \exp(-F_p^2/2\sigma_s^2) \quad (10)$$

$$\widetilde{D}(n_p, \text{sink}) = 1 - \widetilde{D}_u(n_p, \text{source}) \quad (11)$$

where σ_s is empirically set to 0.05 in all our experiments. Intuitively speaking, if a pixel p has low smoothness value F_p, $\widetilde{D}(n_p, \text{source})$ should be relatively high, and there is a high probability that n_p will be connected to the source. Similarly, for every pair of (4-connected) neighboring pixels p and q in the image, we connect n_p and n_q with an edge. The smoothness cost associated with each edge connecting two non-terminal nodes measures how likely the two neighboring pixels are to have the same label. We connect the nodes for every pair of nieghboring pixels p and q by an edge, whose associated cost is

$$\widetilde{S}(n_p, n_q) = 1 - |J_p - J_q| \quad (12)$$

Intuitively, if the intensity values of two neighboring pixels are similar in the smoothed image, there is a high probability that they are in the same texture region. Finally, we solve this graph-cut problem by minimizing the following energy function:

$$\sum_{p,u} \widetilde{D}(n_p, u) + w_{\text{v}} \sum_{p,q} \widetilde{S}(n_p, n_q) \quad (13)$$

where $u \in \{\text{source}, \text{sink}\}$ is the label, and w_{v} weights data and smoothness costs. We empirically set w_{v} to 0.25 in all our experiments. After graph-cut, pixels assigned to source form the final texture region.

5.3 User control

Given a set of user-specified strokes, while our system quite stably extracts texture regions based on a set of pre-defined parameters, we also allow user control. We let the user control the final region by adjusting a single parameter $z \in [-1, 1]$. The smaller z, the smaller the final texture region will be, and vice versa. We achieve this by incorporating z in the graph-cut formulation; in particular, the data cost is re-defined as

$$\widetilde{D}_u(n_p, \text{source}) = P(\exp(-F_p^2/\sigma_v) + z, 0, 1) \quad (14)$$

$$\widetilde{D}(n_p, \text{sink}) = 1 - \widetilde{D}_u(n_p, \text{source}) \quad (15)$$

where $P(v, c_1, c_2)$ is a piecewise function defined as

$$P(v, c_1, c_2) = \begin{cases} c_1, & v \leqslant c_1 \\ v, & c_1 < v < c_2 \\ c_2, & v \geqslant c_2 \end{cases} \quad (16)$$

If z is set to -1, all pixels are labeled as sink and the extracted region is empty. If z is set to 1, all pixels are labeled as source and the extracted region is the whole image. The default value of z is 0. We show an example of parameter tuning in Fig. 8. Even though user can control the extracted region with this parameter, our method is quite stable. In fact, the extracted region is constant for $z \in [-0.6, 0.6]$ in this case.

6 Results and discussion

6.1 Validation

To validate the effectiveness of our method, we have applied our methods to manga images with a variety of texture patterns, including regular patterns (e.g., as in Fig. 9), near-regular patterns (e.g., as in Fig. 10), and irregular patterns (e.g., as in Figs. 5 and 11). We also compare our results with two state-of-the-art methods tailored for manga colorization [6] and manga vectorization [15] respectively.

Figure 9(a) shows a manga image of a cat with a dot pattern. While the feature-based method [6] failed to detect precise boundaries of the texture regions (see Fig. 9(b)), the primitive-based method [15] correctly detects the regions by formulating a specific dot pattern model (see Fig. 9(c)). However, Yao et al.'s method makes very strong assumptions about the primitives in the textures, so they can only handle well textures such as dots, stripes, and grids. In comparison, we make no assumption about the primitives in the textures, but our method can still achieve similar results to those in Ref. [15] by use of texture feature analysis and smoothness diffusion (see Fig. 9(d)). Figure 10 shows another comparison between the results of Ref. [6] and our method. While both their method and ours analyze the texture of the user-specified

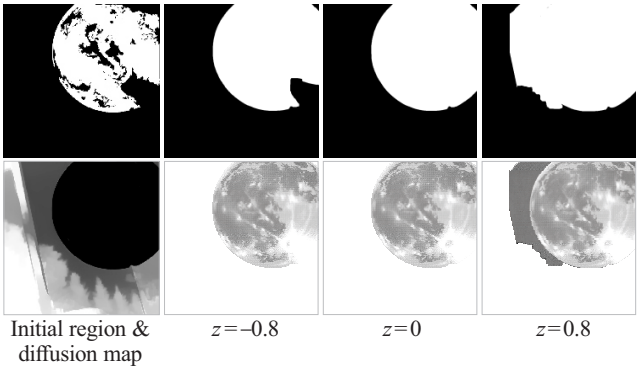


Fig. 8 User control of the extracted region via a single parameter z.

(a) Input (b) Qu et al. [6] (c) Yao et al. [15] (d) Our result

Fig. 9 'Cat' (956 × 943, loading time: 3.60 s, processing time: 3.86 s).

(a) Input (b) Qu et al. [6] (c) Our result

Fig. 10 'Boy' (1712 × 907, loading time: 9.72 s, processing time: 2.07 s).

(a) Input (b) Texture region masks (c) Segmented regions

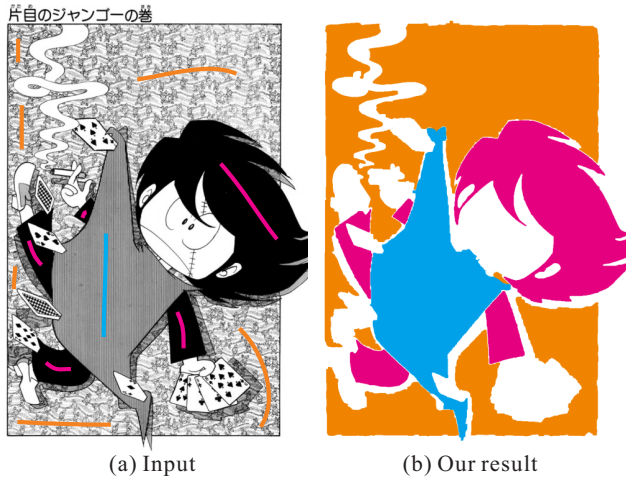
Fig. 11 'Cars' (762 × 843, loading time: 4.03 s, processing time: 1.38 s).

stroke, their method only uses a single texture feature to represent the whole stroke. Therefore, their method is incapable of finding texture regions that are spatial-varying (see Fig. 10(b)). In contrast, our method can handle spatial-varying textures well (see Fig. 10(c)).

In Figs. 5 and 11 the user specifies texture regions with large spatial variation, especially in Fig. 11. By using a set of texture feature vectors to represent the user-specified strokes, our method successfully extracts the texture regions with precise boundaries. Since a manga image may contain multiple textures, we also allow user to specify several sets of strokes indicating different texture regions. The user-specified strokes are sequentially processed so that the user can control the extracted regions more easily. We show three examples in Figs. 12–14 where each input image contains multiple different textures including solid-color regions, regular texture regions, and irregular texture regions. Our method achieves good results in all cases.

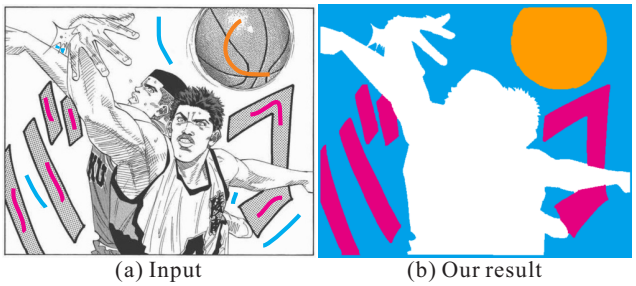
6.2 Timing statistics

All of our experiments were conducted on a PC with a 2.7 GHz CPU and 64 GB memory; all tests were single threaded, unoptimized code, and no GPU was used. We break down the computational time for each example into two parts, the loading time and the processing time (given in the caption of each figure). The loading time is the time spent



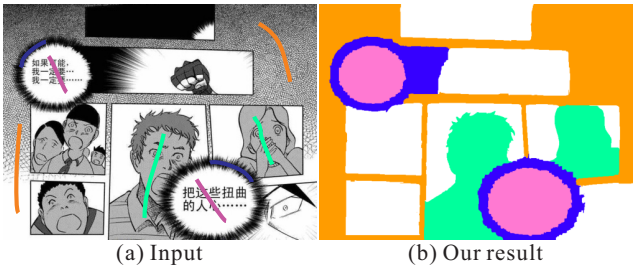
(a) Input (b) Our result

Fig. 12 'Poker' (838 × 1210, loading time: 7.05 s, processing time: 4.72 s).



(a) Input (b) Our result

Fig. 13 'Basketball' (1251 × 1013, loading time: 7.97 s, processing time: 6.27 s).



(a) Input (b) Our result

Fig. 14 'Astonished' (1251 × 1013, loading time: 7.10 s, processing time: 5.36 s).

immediately when the user loads an image into the system, and can be regarded as the offline computation time. The reaction time is the time spent when the system returns the extracted region after the strokes have been drawn, and can be regarded as the online computation time. Whenever the user draws a new stroke or adjusts the control parameter, only the online parts need to be re-executed. We observe that total computation time depends strongly on the resolution of the input image.

6.3 Limitations

One of our limitations concerns the latent assumption that the boundaries between two regions are sharp and smooth. Currently, we cannot handle blurred boundaries well. Furthermore, if the boundary of a region is quite spiky (e.g., the shock balloons in Fig. 14(a)), our current graph-cut formulation will result in a smoothed boundary (e.g., blue regions in Fig. 14(b)). Our method also cannot separate neighboring regions if they are visually inseparable. For example, in Fig. 14(a), the black boundary of the top left shock balloon is connected to the black background of the second panel. Furthermore, the boundary in the smoothed image may deviate by one or two pixels from the original boundary due to limitations of the texture smoothing technique. In this case, we may also fail to extract the precise boundaries of the texture regions.

7 Conclusions

In this paper, we have proposed a novel system to extract texture regions with precise boundaries. Our method starts from an input image and a set of user-specified strokes, and extracts initial regions containing pixels with similar textures, using Gabor wavelets. However, texture features, such as Gabor wavelets, cannot provide precise boundaries. We further smooth the original image via a texture smoothing technique, and refine the initial regions based on the smoothed image. Our method outperforms existing methods in extracting precise boundaries especially for spatial-varying textures.

While our method currently assumes hard boundaries, we could adopt matting techniques instead of the current graph-cut formulation to restore regions with alpha values. We also note that identification of regions depends highly on the semantics of the image content, and introducing perception-based edge extraction techniques could help extract more precise boundaries.

Acknowledgements

This project was supported by the National Natural Science Foundation of China (Project No. 61272293),

and Research Grants Council of the Hong Kong Special Administrative Region under RGC General Research Fund (Project Nos. CUHK14200915 and CUHK14217516).

References

[1] Tuytelaars, T.; Mikolajczyk, K. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision* Vol. 3, No. 3, 177–280, 2008.

[2] Xu, L.; Yan, Q.; Xia, Y.; Jia, J. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 139, 2012.

[3] Julesz, B. Textons, the elements of texture perception, and their interactions. *Nature* Vol. 290, 91–97, 1981.

[4] Weldon, T. P.; Higgins, W. E.; Dunn, D. F. Efficient Gabor filter design for texture segmentation. *Pattern Recognition* Vol. 29, No. 12, 2005–2015, 1996.

[5] Varma, M.; Zisserman, A. Texture classification: Are filter banks necessary? In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, II-691-8, 2003.

[6] Qu, Y.; Wong, T.-T.; Heng, P.-A. Manga colorization. *ACM Transactions on Graphics* Vol. 25, No. 3, 1214–1220, 2006.

[7] Hofmann, T.; Puzicha, J.; Buhmann, J. M. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 20, No. 8, 803–818, 1998.

[8] Paragios, N.; Deriche, R. Geodesic active regions for supervised texture segmentation. In: Proceedings of the 7th IEEE International Conference on Computer Vision, Vol. 2, 926–932, 1999.

[9] Hays, J.; Leordeanu, M.; Efros, A. A.; Liu, Y. Discovering texture regularity as a higher-order correspondence problem. In: *Computer Vision–ECCV 2006*. Leonardis, A.; Bischof, H.; Pinz, A. Eds. Springer Berlin Heidelberg, 522–535, 2006.

[10] Liu, Y.; Collins, R. T.; Tsin, Y. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 26, No. 3, 354–371, 2004.

[11] Liu, Y.; Lin, W.-C.; Hays, J. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics* Vol. 23, No. 3, 368–376, 2004.

[12] Liuy, Y.; Belkina, T.; Hays, J. H.; Lublinerman, R. Image de-fencing. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1–8, 2008.

[13] Siddiqui, H.; Boutin, M.; Bouman, C. A. Hardware-friendly descreening. *IEEE Transactions on Image Processing* Vol. 19, No. 3, 746–757, 2010.

[14] Kopf, J.; Lischinski, D. Digital reconstruction of halftoned color comics. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 140, 2012.

[15] Yao, C.-Y.; Hung, S.-H.; Li, G.-W.; Chen, I.-Y.; Adhitya, R.; Lai, Y.-C. Manga vectorization and manipulation with procedural simple screentone. *IEEE Transactions on Visualization and Computer Graphics* Vol. 23, No. 2, 1070–1084, 2017.

[16] Aujol, J.-F.; Gilboa, G.; Chan, T.; Osher, S. Structure-texture image decomposition—Modeling, algorithms, and parameter selection. *International Journal of Computer Vision* Vol. 67, No. 1, 111–136, 2006.

[17] Meyer, Y. *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations: The Fifteenth Dean Jacqueline B. Lewis Memorial Lectures*. American Mathematical Society, 2001.

[18] Rudin, L. I.; Osher, S.; Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* Vol. 60, Nos. 1–4, 259–268, 1992.

[19] Yin, W.; Goldfarb, D.; Osher, S. Image cartoon-texture decomposition and feature selection using the total variation regularized L^1 functional. In: *Variational, Geometric, and Level Set Methods in Computer Vision*. Paragios, N.; Faugeras, O.; Chan, T.; Schnörr, C. Eds. Springer Berlin Heidelberg, 73–84, 2005.

[20] Durand, F.; Dorsey, J. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics* Vol. 21, No. 3, 257–266, 2002.

[21] Fattal, R.; Agrawala, M.; Rusinkiewicz, S. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics* Vol. 26, No. 3, Article No. 51, 2007.

[22] Paris, S.; Durand, F. A fast approximation of the bilateral filter using a signal processing approach. In: *Computer Vision–ECCV 2006*. Leonardis, A.; Bischof, H.; Pinz, A. Eds. Springer Berlin Heidelberg, 568–580, 2006.

[23] Kass, M.; Solomon, J. Smoothed local histogram filters. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 100, 2010.

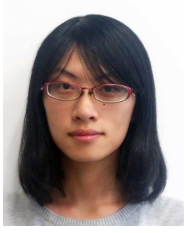
[24] Farbman, Z.; Fattal, R.; Lischinski, D.; Szeliski, R. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics* Vol. 27, No. 3, Article No. 67, 2008.

[25] Subr, K.; Soler, C.; Durand, F. Edge-preserving multiscale image decomposition based on local extrema. *ACM Transactions on Graphics* Vol. 28, No. 5, Article No. 147, 2009.

[26] Xu, L.; Lu, C.; Xu, Y.; Jia, J. Image smoothing via L_0 gradient minimization. *ACM Transactions on Graphics* Vol. 30, No. 6, Article No. 174, 2011.

[27] Manjunath, B. S.; Ma, W.-Y. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 18, No. 8, 837–842, 1996.

Xueting Liu received her B.Eng. degree from Tsinghua University and Ph.D. degree from the Chinese University of Hong Kong in 2009 and 2014, respectively. She is currently a postdoctoral research fellow in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. Her research interests include computer graphics, computer vision, computational manga and anime, and non-photorealistic rendering.



Chengze Li received his B.S. degree from University of Science and Technology of China in 2013. He is currently a Ph.D. student in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. His research interests include computer vision, pattern recognition, and high-performance computing.

Tien-Tsin Wong received his B.Sc., M.Phil., and Ph.D. degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. He is currently a professor in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. His main research interests include computer graphics, computational manga, precomputed lighting, image-based rendering, GPU techniques, medical visualization, multimedia compression, and computer vision. He received the *IEEE Transactions on Multimedia* Prize Paper Award 2005 and the Young Researcher Award 2004.